Università di Pisa

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Implementazione del sistema di protezione contro Meltdown nel nucleo didattico

Candidato:
Riccardo Sagramoni
Matricola 565472

Relatore:
Ing. Giuseppe Lettieri

Anno Accademico 2019-2020

Indice

Capitolo 1

Introduzione

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar

lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Capitolo 2

Meltdown e il sistema di protezione KAISER

La sicurezza dei sistemi informatici attuali si fonda sull'isolamento della memoria, ad esempio marcando come privilegiati gli indirizzi di memoria kernel e bloccando eventuali accessi da parte di programmi utente [6]. **Meltdown** è un tipo di attacco informatico che sfrutta un effetto collaterale dell'esecuzione fuori ordine nei processori moderni per leggere locazioni di memoria scelte in maniera arbitraria. L'attacco funziona su varie microarchitetture Intel prodotte sin dal 2010, indipendentemente dal sistema operativo in uso. Meltdown è quindi in grado di accedere arbitrariamente a qualsiasi locazione di memoria protetta (afferenti al kernel o ad altri processi) senza necessitare alcun permesso o privilegio da parte del sistema [7].

Meltdown rompe quindi tutti i meccanismi di sicurezza che si basano sull'isolamento degli spazi di indirizzamento, andando a colpire milioni di utenti. Il sistema di protezione KAISER, sviluppato originariamente per KASLR [2], ha l'importante effetto secondario di impedire l'utilizzo di Meltdown [7].

2.1 Background

2.1.1 Esecuzione Fuori Ordine

2.1.2 Spazi di indirizzamento

Per risolvere diversi problemi, in particolare l'isolamento dei processi [4], le CPU supportano l'utilizzo di spazi d'indirizzamento virtuali, in cui gli indirizzi virtuali (relativi al singolo processo) vengono tradotti in indirizzi fisici. Lo

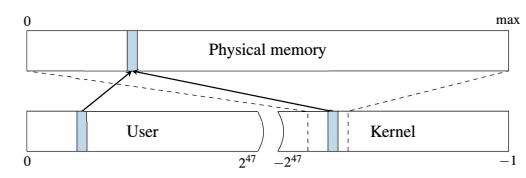

Figura 2.1: text

spazio d'indirizzamento di un processo (ovvero tutti i possibili indirizzi che un processo può generare) viene suddiviso in regioni dette *pagine* che possono essere mappate individualmente nella memoria fisica attraverso una tabella di traduzione multivello. Ogni processo possiede una propria tabella di traduzione che traduce tutti e soli i suoi indirizzi virtuali e che definisce le proprietà di protezione delle varie zone di memoria.

Ogni processo può quindi riferirsi esclusivamente agli indirizzi appartenenti al proprio spazio di indirizzamento virtuale. Per permettere l'utilizzo

2.1.3 Attacchi Cache

Al fine di velocizzare gli accessi alla RAM, le CPU contengono buffer di memoria molto veloce ma di dimensioni limitate che costituiscono la cosiddetta *memoria cache*. La memoria cache maschera i tempi di latenza estremamente lunghi per l'accesso alla memoria centrale (molto lenta in confronto alla cache) conservando le locazioni di memoria che, secondo principi statistici come la *località spaziale* (se un programma accede ad un certo indirizzo, è molto probabile che in breve tempo accederà ad un indirizzo vicino) e la *località temporale* (se un programma accede ad un certo indirizzo, è molto probabile che in breve tempo vi accederà di nuovo), è più probabile vengano indirizzate dalla CPU nel breve periodo [3].

Gli attacchi a canale laterale (*side-channel attacks*) contro la cache sfruttano questa differenza di tempo di accesso introdotta dalla cache stessa. Negli attacchi Flush+Reload [8], usati da Meltdown [7], l'attaccante è in grado di determinare se una locazione di memoria è stata precedentemente caricata in cache, misurando il tempo impiegato da un'operazione di lettura.

2.2 Come agisce Meltdown

L'attacco Meltdown consiste in due fasi fondamentali [7]:

1. Far eseguire alla CPU una o più istruzioni in maniera speculativa (le *transient instruction*) che accedono a locazioni di memoria **protetta** (kernel o altri processi).

2. Leggere gli effetti dell'esecuzione delle transient instruction a livello di microarchitettura per ottenere il contenuto delle locazioni di memoria accedute.

2.2.1 L'esecuzione delle transient instruction

Nella prima fase di Meltdown, l'attaccante cerca di accedere ad una zona di memoria protetta, ad esempio la memoria kernel. Il tentativo di accesso ad una pagina non accessibile da livello utente fa in modo che la CPU sollevi un'eccezione di protezione, che generalmente termina il processo. Tuttavia, a causa dell'esecuzione fuori ordine, la CPU potrebbe aver già eseguito l'istruzione di accesso in maniera speculativa *prima* delle istruzioni relative all'eccezione di protezione, al fine di minimizzare i tempi di latenza (vedi paragrafo 2.1.1). Grazie al lancio dell'eccezione, le eventuali istruzioni eseguite in maniera speculativa, relative dunque ad una previsione di salto *errata*, non vengono ritirate dalla CPU e non hanno così alcun effetto sulla macroarchitettura in generale (memoria centrale e registri logici non speculativi del processore) [1].

Nonostante non si verificano effetti macroarchitetturali, le transient instruction hanno effetti secondari a livello di **microarchitettura**. Durante l'esecuzione speculativa, la locazione di memoria riferita viene memorizzata sia nella cache che in un registro fisico della CPU. Sebbene il loro contenuto non venga mai reso disponibile al programma utente grazie al mancato ritiro delle micro-operazioni, la locazione di memoria resta in cache, diventando possibile vittima di un attacco side-channel alla memoria cache [7].

Come abbiamo detto nel paragrafo 2.1.2, l'intera memoria fisica viene mappata all'interno dello spazio di indirizzamento del kernel attraversa la cosiddetta *finestra di memoria fisica* [5].

2.2.2 title

Capitolo 3

Introduzione al nucleo didattico

Capitolo 4

L'implementazione del sistema di protezione

Ciao [**Lipp:meltdown**][**Gruss:kaslr**]

Bibliografia

[1] Graziano Frosini e Giuseppe Lettieri. *Architettura dei calcolatori Vol. II*. A cura di Pisa University Press. 2013.

[2] Daniel Gruss et al. «KASLR is Dead: Long Live KASLR». In: *International Symposium on Engineering Secure Software and Systems*. Springer International Publishing, 2017, pp. 161–176.

[3] Giuseppe Lettieri. *Memoria Cache*. 16 Mar. 2017. URL: `https://calcolatori.iet.unipi.it/resources/cache.pdf`.

[4] Giuseppe Lettieri. *Paginazione*. 3 Mag. 2019. URL: `https://calcolatori.iet.unipi.it/resources/paginazione.pdf`.

[5] Giuseppe Lettieri. *Paginazione: complementi*. 4 Apr. 2017. URL: `https://calcolatori.iet.unipi.it/resources/tlb.pdf`.

[6] Giuseppe Lettieri. *Protezione*. 11 Apr. 2019. URL: `https://calcolatori.iet.unipi.it/resources/protezione.pdf`.

[7] Moritz Lipp et al. «Meltdown: Reading Kernel Memory from User Space». In: *27th USENIX Security Symposium (USENIX Security 18)*. 2018.

[8] Yuval Yarom e Katrina Falkner. «FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack». In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 719–732.